UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de:

Ingeniero de Sistemas

TEMA:
DESARROLLO DE UNA APLICACIÓN MÓVIL ANDROID PARA LA CONSULTA DE RUTAS DE UNA LÍNEA DE BUSES URBANOS QUE CIRCULAN POR LA CIUDAD DE QUITO REFERENCIANDO LOS PUNTOS DE PARTIDA Y DESTINO DEL USUARIO.

AUTORES:
JONATHAN FERNANDO ALMEIDA MUÑOZ
SEBASTIÁN MARTÍN SOLÍS CUÑEZ

TUTOR:
ALONSO RENÉ ARÉVALO CAMPOS

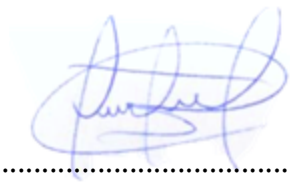
Quito, febrero del 2019

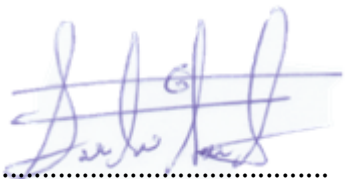
CESIÓN DE DERECHOS DE AUTOR

Nosotros, Jonathan Fernando Almeida Muñoz, con CI. 172165025-5 y Sebastián Martín Solís Cuñez, con CI. 172288923-3, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación con el tema: "DESARROLLO DE UNA APLICACIÓN MÓVIL ANDROID PARA LA CONSULTA DE RUTAS DE UNA LÍNEA DE BUSES URBANOS QUE CIRCULAN POR LA CIUDAD DE QUITO REFERENCIANDO LOS PUNTOS DE PARTIDA Y DESTINO DEL USUARIO", mismo que ha sido desarrollado para optar por el título de INGENIEROS DE SISTEMAS en la universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada.

En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.





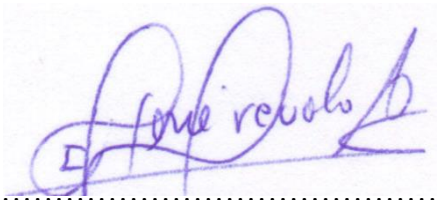
.....	.....
JONATHAN FERNANDO	SEBASTIÁN MARTÍN
ALMEIDA MUÑOZ	SOLÍS CUÑEZ
CI: 1721650255	CI: 1722889233

Quito, febrero del 2019

DECLARATORIA DE COAUTORÍA DEL TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Proyecto Técnico, con el tema: DESARROLLO DE UNA APLICACIÓN MÓVIL ANDROID PARA LA CONSULTA DE RUTAS DE UNA LÍNEA DE BUSES URBANOS QUE CIRCULAN POR LA CIUDAD DE QUITO REFERENCIANDO LOS PUNTOS DE PARTIDA Y DESTINO DEL USUARIO realizado por Jonathan Fernando Almeida Muñoz y Sebastián Martín Solís Cuñez, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, febrero del 2019



……..

ALONSO RENÉ ARÉVALO CAMPOS

CI: 1400164891

Dedicatoria

Dedicamos este proyecto a nuestros padres y hermanos que nos brindaron el apoyo, amor y comprensión durante toda nuestra formación profesional siendo un pilar fundamental que nos dio la fuerza necesaria para poder culminar con éxito nuestro objetivo

Jonathan Fernando Almeida Muñoz

Sebastián Martín Solís Cuñez

Agradecimientos

Agradezco a la Universidad Politécnica Salesiana por darnos la oportunidad de tener una formación profesional y personal a través de todos los ingenieros aportando hacia nosotros grandes conocimientos y valores.

También a nuestro tutor del proyecto de titulación el Ingeniero René Arévalo que confió en nuestras capacidades para desarrollar este proyecto

Jonathan Fernando Almeida Muñoz

Sebastián Martín Solís Cuñez

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El proyecto tiene como propósito ayudar a los ciudadanos de la ciudad de Quito a llegar a su destino de una forma clara e intuitiva, visualizando las rutas de la flota de buses que se encuentran cerca de su ubicación, únicamente ingresando como parámetro el lugar de destino. Mostrando al usuario cual es la ruta determinada, el código, y el nombre de la cooperativa de transporte. De esta manera se reducirá la pérdida de tiempo por una mala planificación o desconocimiento de las rutas al destino.

A través de la metodología ágil scrum se lleva a cabo las tareas planteadas de forma rápida y con un margen para cambios pedidos por el usuario.

La aplicación está basada en la arquitectura cliente-servidor, teniendo en el lado del cliente la aplicación móvil, y en el servidor un entorno basado en un api web que a través de peticiones REST permite interactuar con el sistema gestor de datos realizando operaciones como modificación, eliminación o inserción de rutas.

El cálculo de las rutas ideales se ejecuta en las funciones del sistema de administración de las bases de datos espaciales, obteniendo las mejores rutas según la distancia recorrida y número de unidades a tomar según el destino ingresado por parte del usuario.

Como resultado se consigue una consistencia al realizar las búsquedas de rutas obteniendo un tiempo promedio de 34 milisegundos (una ruta) y de 51 segundos (dos rutas), para 500 usuarios concurrentes. Lo que permite a los usuarios un servicio útil y confiable.

ABSTRACT

The purpose of the project is to help the citizens of Quito's city reach their destination in a clear and intuitive way, visualizing the routes of the bus that operate in Quito and are close to their location, as entered as a parameter the place of destination, the code and the name of the transport line. This will reduce time due to poor planning and ignorance of the routes to the destination.

Using the agile scrum methodology to carry out the tasks quickly with a margin for changes requested by the user.

The application is based on the server - client architecture, on the client side, the mobile application, and in the server side with an environment based on a web interface that, through REST requests, allows the interaction with the base engine. Operations data as modification, elimination and insertion of routes.

The estimation of the ideal routes is done through the functions of the administration system of the spatial database (Postgres - Postgis), obtaining the best routes according to the distance and the number of units according to the destination entered by the user.

The consistency achieved searching for routes, obtaining an average time of 34 milliseconds (for one route) and 51 seconds (for two routes), using 500 concurrent users. Is a result that allows users a useful and reliable service.

INTRODUCCIÓN

El desconocimiento de los usuarios sobre las rutas que realizan las flotas de buses en la ciudad, el intervalo en el que las unidades salen de su despacho, los horarios de operación de las unidades de transporte son las causas que provocan al usuario un retraso en sus tiempos de movilización, o incluso perdiéndose al tomar rutas equivocadas que no lleven al destino planificado.

Este problema se agrava cuando las personas arriban de provincias u otros países, hacia la ciudad de Quito ya que al no tener conocimiento de que unidad tomar para arribar a su destino, llegando a una única opción de hacer uso del servicio de taxis que tiene un valor monetario más elevado en comparación al uso del servicio público.

Visualizar las rutas disponibles que actualmente operan en el distrito metropolitano desde sus dispositivos móviles de forma fácil e intuitiva fomentará el conocimiento entre los ciudadanos para que se puedan dirigir desde un lugar a otro en los buses de servicio público.

Este proyecto técnico tiene con objetivo informar sobre las rutas de los buses de Quito las cuales les servirá para movilizarse desde un lugar de origen a su destino, tomando en cuenta las paradas más cercanas y lugares cercanos para determinar la ruta más adecuada del usuario de la aplicación móvil.

La movilidad humana es un ámbito fundamental para el desarrollo de las ciudades principalmente en el Distrito Metropolitano de Quito que al ser una ciudad ubicada entre montañas y siendo la única posibilidad de expandirse de forma lineal, la movilidad en transporte público en la ciudad es uno de los mayores problemas, debido a que no tiene se ha desarrollado un plan vial adecuado y cuya capacidad está siendo superada cada vez más y para poder solventar este inconveniente se han venido

integrando más rutas al sistema de movilidad, siendo este el principal inconveniente para las personas que visitan la ciudad e incluso a las personas que no usan habitualmente el sistema de transporte público.

Se han realizado intentos para informar a los usuarios sobre las rutas que operan las diferentes cooperativas de transporte público, como por ejemplo la compañía de transporte urbano del sur Disutran SA, despliega en su página web sus servicios así como también las diferentes rutas que realiza su flota de buses a través de google maps, mostrando las rutas y sectores del recorrido de forma gráfica señalando mediante líneas manualmente la ruta.

Estas aproximaciones no son amigables para el usuario, por su difícil acceso, uso e interacción con las diferentes líneas de transporte público de Quito.

También se puede ver la evolución en los servicios de taxis con aplicaciones como uber, cabify o easy taxi, que dan un valor al usuario por su facilidad de uso al contactar con los taxistas, mostrando gráficamente la ubicación de los involucrados. Siendo un claro caso de éxito en el que la tecnología móvil aporta para mejorar la movilidad de los usuarios.

Estos conceptos aplicados al servicio de transporte público de la ciudad aportan a una mayor población.

Problema

La ciudad de Quito por ser la capital del Ecuador y un destino apetecido por turistas nacionales y extranjeros, genera una gran demanda en el uso del transporte público, teniendo como desventaja la poca difusión del uso de buses como medio de transporte.

La movilización en el Distrito Metropolitano de Quito en transporte público, siempre ha sido confusa ya que por la ubicación geográfica, la extensión territorial de la capital

del Ecuador y la existencia de más de 200 líneas de transporte han logrado confabular para que los ciudadanos tengan lugar para confusiones, retrasos e incluso pérdidas dentro del perímetro urbano de la ciudad.

El desconocimiento de los usuarios sobre las rutas que realizan las flotas de buses en la ciudad, el intervalo en el que las unidades salen de su despacho, los horarios de operación de las unidades de transporte, son las causas que provocan al usuario un retraso en sus tiempos de movilización, o incluso perdiéndose al tomar rutas equivocadas que no lleven al destino planificado. Este problema se agrava cuando las personas arriban de provincias u otros países, hacia la ciudad de Quito por no tener conocimiento que unidad tomar para llegar a su destino, llegando a una única opción de hacer uso del servicio de taxis que tiene un valor económico más elevado en comparación a la movilización en buses urbanos.

El desarrollo de una aplicación móvil es un proyecto de alta importancia para ayudar a los ciudadanos de la ciudad de Quito a llegar a su destino de una forma clara e intuitiva, visualizando las rutas de la flota de buses que se encuentran cerca de su ubicación, únicamente ingresando como parámetro el lugar de destino. Mostrando al usuario cual es la ruta determinada, el código de la misma, y el nombre de la cooperativa de transporte. De esta manera se disminuiría la pérdida de tiempo por una mala planificación y desconocimiento de las rutas hacia el destino ingresado.

Con la funcionalidad de la aplicación, se apoyará a la reducción de tiempos en la movilización de usuarios así como también ahorrando dinero ya que al tener comprensión de las rutas, los usuarios tomarán menos buses. Llevando a que se pueda escalar la aplicación, aumentando las flotas de buses con sus respectivas líneas, para

que en un futuro sea una solución integral para todos los usuarios del servicio de buses de la ciudad.

Objetivo General

Desarrollar una aplicación móvil para dispositivos que funcionan con sistema operativo Android en la cual se podrá visualizar las líneas urbanas de buses que circulan por un punto georreferenciado en la ciudad de Quito.

Objetivos Específicos

Georreferenciar el punto de ubicación del usuario de la línea de transporte público.

Diseñar un sistema web de fácil manejo y amigable para el usuario, utilizando la metodología SCRUM.

Desarrollar una aplicación móvil Android para el despliegue de las líneas de buses urbanos que pasen por el punto georreferenciado del usuario.

Visualizar en un mapa la ruta de la línea de buses, que seleccione el posible usuario o pasajero.

Implementación de un módulo para añadir, editar y eliminar rutas para tener actualizada la información de coordenadas y rutas.

Capítulo 1

Marco teórico

1.1. Recolección de información

El proceso de recolección de información de las rutas que atraviesan la ciudad de Quito incluyendo sus respectivas paradas, circuitos, intervalos de cada una de las líneas que operan dentro de la ciudad se lo realizó directamente con el ente encargado del tema, como lo es la Secretaria De Movilidad, para comenzar el proceso de recolección de información se envió un oficio con los datos de los estudiantes dirigido al Sr. Abg. Rubén Darío Tapia – Secretario de Movilidad. Solicitando al área de tecnología los archivos con líneas, puntos, polilíneas georreferenciadas de la ciudad de Quito para la realización del presente proyecto.

1.2. Metodologías ágiles

Hablar de las metodologías ágiles involucra hacer referencia a todas las consideradas metodologías de desarrollo de software tradicionales ya que desde sus inicios se promovió al origen del uso de metodologías ágiles; sus características principales son que tienen un funcionamiento diferente. Para lo cual se define Metodologías Ágiles como: "Las metodologías ágiles son flexibles, pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto."(Canós, 2012)

Las metodologías ágiles principalmente ayudan a optimizar los tiempos de una manera efectiva, así como también permiten realizar las tareas pendientes de manera clara y concisa pudiendo adaptarse a los tiempos del equipo de desarrollo.

1.3. Diagrama de Casos de Uso

Simula la interacción de un cliente real con el sistema y se puede detallar como operan los elementos del sistema. (Larman, 2003)

1.4. Diagrama de Secuencia

Se crean a partir de los diferentes diagramas de casos de uso, una vez que se tiene identificado a todos los actores del sistema, aquí se refleja el intercambio de mensajes, métodos, información con la que interactuaran los sistemas. (Booch, 1999)

1.5. Tecnologías

1.5.1. API

Interfaz de programación de aplicaciones, permite principalmente ejecutar funciones y procedimientos propios del proyecto, se encuentra en la capa de abstracción. Al utilizar una determinada API se tiene que obligatoriamente usar las funciones desarrolladas para ese proyecto, teniendo la limitación de no desarrollar más funciones ya que se encuentran encapsuladas para evitar el desarrollo por terceros. (Doglio, 2015)

1.5.2. Servicios Web

Utiliza un sinfín de protocolos que sirven principalmente para intercambiar datos entre las diferentes aplicaciones, sin importar el IDE de desarrollo, ni entorno de ejecución, ya que interactúan con todos servicios de maquina en maquina mediante una conexión de red/ internet. (IBM, s.f.)

1.5.3. REST

Es un tipo de tecnología que utiliza cualquier interfaz entre sistemas usando el hypertext transfer protocol para poder generar aplicaciones de datos y operaciones para

su posterior procesamiento y entendimiento en formatos XML, JSON. Atendiendo siempre mediante una dirección URL y un puerto especifico. (Open, 2019)

1.5.4. Android

Android es un sistema operativo desarrollado principalmente para aparatos móviles en los que incluye un sistema operativo, middleware y una infinidad de aplicaciones para su funcionamiento. Basado principalmente en el kernel de Linux y adaptable a la mayoría de pantallas lo convierte en uno de los sistemas operativos más usados de todos los tiempos. (Gironés, 2012)

1.5.5. Shapefile SHP

Es un formato vectorial de almacenamiento digital en donde se guardan las ubicaciones geográficas con sus atributos de cada una de las geometrías, se pueden representar por medio de puntos o líneas. (Chang, 2006)

1.5.6. APK

Un APK es un archivo ejecutable de una aplicación desarrollada para el S.O. Android, es muy parecido al formato EXE - Windows, este tipo de archivo contiene todos los componentes para ejecutar una aplicación. (Gironés, 2012)

1.5.7. WGS 84

Sistema Geodésico Mundial 1984 permite representar cualquier punto geográfico de la tierra basado en tres tipos de unidades, XYZ.

1.5.8. PostgreSQL

Es un motor de bases de datos normalizada y relacional, se distribuye a través de una licencia Open Source puede ser utilizado, distribuido y modificado por cualquier persona y es utilizado para cualquier propósito ya sea comercial, empresarial, educativo. Para el desarrollo de bases de datos espaciales es compatible con su

complemento Postgis ofreciendo velocidad en el procesamiento de coordenadas, geométricas y objetos. (Connolly, 2005)

1.5.9. Bases de datos espaciales

Se usan principalmente para almacenar coordenadas, geometrías, puntos georreferenciados. Su uso se aplica para gestionar los sistemas de información geográfica teniendo los datos bien estructurados, integridad de los datos y de fácil búsqueda de indexación. (Techlandia, 2019)

1.5.10. JavaScript

JavaScript es considerado como un lenguaje de programación orientado a objetos (POO), programado en C, C++. Actualmente es el más utilizado en navegadores para interactuar con clientes mostrando información dinámica como mapas, animaciones, etc. Permitiendo operar entre diferentes entornos de trabajo. (Flanagan, 2006)

1.5.11. Node JS

Node JS es un framework para programación en código abierto que se basa en eventos, está formulado para generar un sistema escalable y consistente soportando de forma simultánea varias conexiones concurrentes, garantiza un elevado rendimiento gracias a su ejecución multihilo, su esencia se basa en el conocido JavaScript pero será ejecutado en lado del servidor. (Cantelon, 2014)

1.5.12. Android Studio

Framework orientado para la programación de aplicaciones móviles exclusivamente para el sistema operativo Android, desarrollado por Google reemplazando así a Eclipse como IDE de desarrollo de aplicaciones móviles. Tiene una licencia Apache 2.0 para ser distribuido y ejecutado de forma gratuita. (Hohensee, 2014)

1.5.13. Quantum GIS

QGIS es un sistema de información geográfica basado en software libre, interactúa con estructuras de datos bidimensionales y tridimensionales de puntos georreferenciados, los cuales son debidamente almacenados en capas y vectores. Con su complemento de PostGis, interactúa directamente con los datos que están almacenados en la base de datos para su posterior visualización con plugins que contienen apis como: Google Maps, OpenStreetMaps, Bing, Apple iphoto map. (Hugentobler, 2008)

1.5.14. PostGis

Postgis es un complemento de Postgres desarrollado por Refractions Research Inc, es una base de datos espacial que almacena información geográfica apoyada en el motor de base de datos Postgres para su posterior almacenamiento de datos en una base de datos de tipo espacial, permite además almacenar diferentes geometrías en una misma estructura para su posterior interacción con la información en la base de datos. (Ramsey, 2005)

1.6. Herramientas de desarrollo

Para el desarrollo de la presente aplicación se necesita de los siguientes componentes:

Figura 1. Herramientas de desarrollo

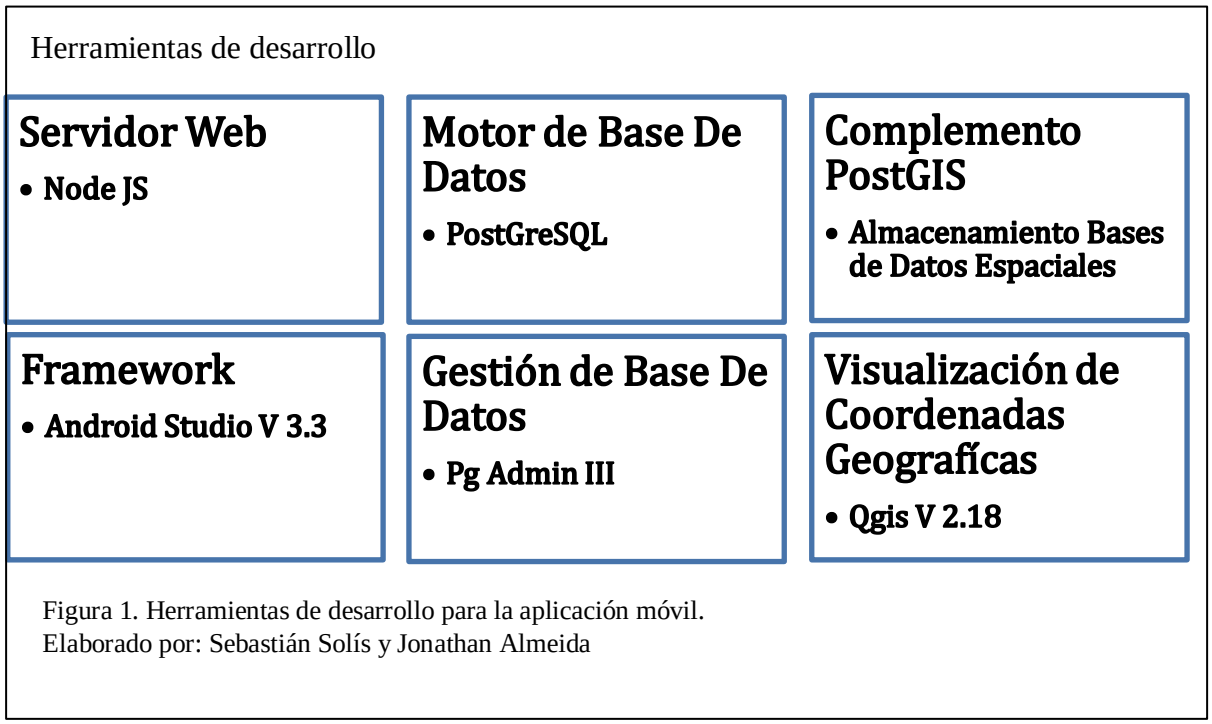


Figura 1. Herramientas de desarrollo para la aplicación móvil.
Elaborado por: Sebastián Solís y Jonathan Almeida

Capítulo 2

Análisis y diseño

2.1 Análisis

La movilidad humana es un pilar clave para el desarrollo de los pueblos y para lograr este objetivo se han desarrollado una serie de facilidades entorno a la movilidad principalmente en el transporte público, cuyo principal inconveniente es la poca difusión de las nuevas alternativas con respecto a la movilidad. El Distrito Metropolitano de Quito al tener bien definido el sistema de transporte público es de gran utilidad para lo cual las distintas herramientas tecnológicas que ayudan a difundir las alternativas de movilidad, para ello con el avance de las aplicaciones móviles se desarrollara una app que ayude a difundir las rutas de transporte público en el Distrito Metropolitano de Quito.

2.2 Marco metodológico

La metodología usada es Scrum, este pertenece a la familia del desarrollo de software con metodologías ágiles el cual se ha venido aplicando en un sin número de proyectos para el desarrollo de software, siendo su principal esencia la administración del desarrollo de software.

La premisa principal de Scrum se basa en que el desarrollo de software es bastante complejo e impredecible como para planearlo con anticipación. (Schwaber, 2002)

2.2.1 Roles Scrum

Los roles de Scrum están definidos para visualizar las funciones desempeñadas por cada actor para el desarrollo de la aplicación móvil. Para lo cual se tienen definido tres tipos de roles:

2.2.2 Scrum máster

Es el líder de proyecto y se le encarga gestionar que los miembros del equipo alcancen sus objetivos hasta llegar al sprint final.

2.2.3 Equipo

Se refiere a todos los integrantes del equipo de desarrollo, los cuales están debidamente capacitados con los conocimientos técnicos para así poner en marcha el proyecto de manera conjunta concluyendo debidamente cada historia e iniciando cada sprint con los resultados propuestos.

2.2.4 StakeHolder

En Scrum un StakeHolder es cualquier persona interesada en el producto, no tiene que ser parte del equipo Scrum

2.2.5 Usuarios del sistema

El sistema para la consulta de rutas cuenta con los siguientes usuarios:

2.2.6 Administrador

Este usuario se le permite realizar las funciones de insertar, actualizar y eliminar rutas en la base de datos por medio del api. Aparte de ejecutar las consultas de rutas.

2.2.7 Usuario registrado

El usuario Invitado solo puede realizar consultas de rutas a través de la aplicación.

2.2.8 Perspectiva del sistema

El servidor web cuenta con los siguientes servicios:

- ✓ Consultar (calcular) rutas a través de la aplicación Android api REST.
- ✓ Insertar rutas a través del api REST.
- ✓ Actualizar rutas a través del api REST.

- ✓ Eliminar rutas a través del api REST.

2.2.9 Requerimientos del sistema

Se muestran los requerimientos no funcionales y funcionales del sistema y las actividades a ser desarrolladas en el transcurso del proyecto, representados en la siguiente tabla con la lista de requerimientos (Product Backlog).

El conjunto de funcionalidades que se van a implementar en el presente proyecto se detalla en la tabla 1.

Tabla 1. Lista de requerimientos (Product Backlog)

Módulo	Iteración	ID HU	Requerimientos	ID Tarea	Tarea	Prioridad	Estimado (Días)	Rol Scrum
Consulta	1	1	Análisis y diseño del sistema	1.1	Diagrama de construcción de procesos	Alta	6	Equipo
				1.2	Diagrama entidad - relación	Media	4	Equipo
		2	Levantamiento del api y autenticación	2.1	Creación servicio web	Media	5	Equipo
				2.2	Desarrollo de autenticación web	Media	4	Equipo
				2.3	Pruebas unitarias	Media	3	Equipo
				2.4	Depuración y optimización	Alta	4	Equipo

		3	Cálculo de rutas	3.1	Funcionamiento api web	Alta	5	Equipo
				3.2	Desarrollo del script de consulta de una ruta	Alta	10	Equipo
				3.3	Desarrollo del script de consulta de dos rutas	Alta	10	Equipo
				3.4	Pruebas unitarias	Media	3	Equipo
				3.5	Depuración y optimización	Media	4	Equipo
Aplicación móvil	2	4	Gestión de aplicación móvil	4.1	Diseño de interfaz móvil	Alta	8	Equipo
				4.2	Desarrollo de la funcionalidad de autenticación	Media	4	Equipo

				4.3	Desarrollo de la búsqueda de direcciones y lugares	Media	6	Equipo
				4.4	Desarrollo de la consulta de rutas al servidor web	Alta	10	Equipo
				4.5	Pruebas unitarias	Media	3	Equipo
				4.6	Depuración y optimización	Alta	4	Equipo
Administración	3	5	Inserción de rutas	5.1	Desarrollo del servicio web	Alta	4	Equipo
				5.2	Desarrollo de la funcionalidad de inserción de rutas	Media	4	Equipo
				5.3	Desarrollo del script de inserción de rutas	Alta	8	Equipo

				5.4	Pruebas unitarias	Media	3	Equipo
				5.4	Depuración y optimización	Media	4	Equipo
	4	6	Eliminación de rutas	6.1	Desarrollo del servicio web	Alta	4	Equipo
				6.2	Desarrollo de la funcionalidad de eliminación de rutas	Media	4	Equipo
				6.3	Desarrollo del script de eliminación de rutas	Alta	8	Equipo
				6.4	Pruebas unitarias	Media	3	Equipo
				6.5	Depuración y optimización	Media	4	Equipo

	5	7	Actualización de rutas	7.1	Desarrollo del servicio web	Alta	4	Equipo
				7.2	Desarrollo de la funcionalidad de actualización de rutas	Media	4	Equipo
				7.3	Desarrollo del script de actualización de rutas	Alta	8	Equipo
				7.4	Pruebas unitarias	Media	3	Equipo
				7.5	Depuración y optimización	Media	4	Equipo

Nota: Esta tabla contiene los requerimientos del sistema con los 5 sprints
Elaborado por: Sebastián Solís y Jonathan Almeida

2.2.10 Pruebas por iteración

Iteración 1: Se encuentra detallado el módulo de consulta el cual contiene las actividades de análisis y diseño del sistema, construcción del api y autenticación web y la consulta de rutas.

Tabla 2. Resultado de la iteración 1

MÓDULO	ITERACIÓN	PRE REQUISITO	ACTOR	PROCESO	SUB-PROCESO	TAREA	RESULTADO ESPERADO	RESULTADO ALCANZADO	OBSERVACIONES
CONSULTA	1	Primera petición para generar la conexión en el servidor.	Administrador/ Usuario Registrado	Levantamiento del api y autenticación	Creación del servicio web	Conexión al Api web mostrando el mensaje de estado de conexión.	Figura anexo 1	OK	
					Autenticación web	Verificar usuario registrado en la base de datos.	Figura anexo 2	OK	
		Autenticación mediante usuario y contraseña	Administrador/ Usuario Registrado	Búsqueda de ruta	Calcular ruta	Obtener la ruta calculada.	Figura anexo 3	Pendiente optimización	Optimización del script para reducir tiempos de procesamiento

Nota: Esta tabla contiene el resultado de la iteración 1
Elaborado por: Sebastián Solís y Jonathan Almeida

Iteración 2: Se encuentra detallado el módulo aplicación móvil el cual contiene las actividades de búsqueda de destino y direcciones haciendo uso del api de google maps y google places. Para su posterior consulta en la base de datos y cálculo de la ruta óptima.

Tabla 3. Resultado de la iteración 2

MÓDULO	ITERACIÓN	PRE REQUISITO	ACTOR	PROCESO	SUB-PROCESO	TAREA	RESULTADO ESPERADO	RESULTADO ALCANZADO	OBSERVACIONES
APLICACIÓN MOVIL	2	Autenticación mediante usuario y contraseña	Administrador/ Usuario Registrado	Búsqueda de direcciones y lugares	Buscar lugares en el api Google Maps	Desarrollo de la búsqueda de lugares.	Figura anexo 4	OK	
					Búsqueda de direcciones usando Marcadores	Desarrollo de la búsqueda de direcciones.	Figura anexo 5	OK	
		Autenticación mediante usuario y contraseña	Administrador/ Usuario Registrado	Sugerencia de rutas del servidor web	Calculo de Rutas	Obtener la o las rutas calculadas.	Figura anexo 6	OK	

Nota: Esta tabla contiene el resultado de la iteración 2
Elaborado por: Sebastián Solís y Jonathan Almeida

Iteración 3: Se encuentra detallado el módulo de administración de rutas el cual contiene las actividades de inserción de rutas.

Tabla 4. Resultado de la iteración 3

MÓDULO	ITERACIÓN	PRE REQUISITO	ACTOR	PROCESO	SUB-PROCESO	TAREA	RESULTADO ESPERADO	RESULTADO ALCANZADO	OBSERVACIONES
ADMINISTRACIÓN	3	Autenticación mediante usuario y contraseña	Administrador	Ingreso de rutas	Conexión con el servicio web	Desarrollo del servicio web	Figura anexo 8	OK	
					Inserción de geometría	Desarrollo de la funcionalidad de inserción de rutas	Figura anexo 9	OK	

Nota: Esta tabla contiene el resultado de la iteración 3
Elaborado por: Sebastián Solís y Jonathan Almeida

Iteración 4: Se encuentra detallado el módulo de administración de rutas el cual contiene las actividades de eliminación de rutas.

Tabla 5. Resultado de la iteración 4

MÓDULO	ITERACIÓN	PRE REQUISITO	ACTOR	PROCESO	SUB-PROCESO	TAREA	RESULTADO ESPERADO	RESULTADO ALCANZADO	OBSERVACIONES
ADMINISTRACIÓN	4	Autenticación mediante usuario y contraseña	Administrador	Eliminación de rutas	Conexión con el servicio web	Desarrollo del servicio web	Figura anexo 10	OK	
					Eliminar geometría	Desarrollo de la funcionalidad de eliminación de rutas	Figura anexo 11	OK	

Nota: Esta tabla contiene el resultado de la iteración 4
Elaborado por: Sebastián Solís y Jonathan Almeida

Iteración 5: Se encuentra detallado el módulo de administración de rutas el cual contiene las actividades de actualización de rutas.

Tabla 6. Resultado de la iteración 5

MÓDULO	ITERACIÓN	PRE REQUISITO	ACTOR	PROCESO	SUB-PROCESO	TAREA	RESULTADO ESPERADO	RESULTADO ALCANZADO	OBSERVACIONES
ADMINISTRACIÓN	5	Autenticación mediante usuario y contraseña	Administrador	Actualización de rutas	Conexión con el servicio web	Desarrollo del servicio web	Figura anexo 12	OK	
					Actualizar geometría	Desarrollo de la funcionalidad de actualización de rutas	Figura anexo 13	Pendiente optimización	Modificación para utilizar n parámetros

Nota: Esta tabla contiene el resultado de la iteración 5
Elaborado por: Sebastián Solís y Jonathan Almeida

2.2.11 Lista de iteraciones (Sprint BackLog)

Durante el desarrollo de software para la aplicación móvil los módulos se han desarrollado conforme avanza el proyecto, para lo cual se debe determinar los tiempos para cada uno de los requerimientos.

Tabla 7. Sprint BackLog

ITERACIÓN	REQUERIMIENTOS	ESTADO	FECHA COMPLETADO	ESTIMADO
1	Análisis y diseño del sistema	COMPLETADO	07/01/2018	65
	Levantamiento del api y autenticación	COMPLETADO	15/01/2018	
	Cálculo de rutas	COMPLETADO	25/02/2018	
2	Gestión de aplicación Móvil	COMPLETADO	28/05/2018	70
3	Inserción de rutas	COMPLETADO	29/08/2018	70
4	Eliminación de rutas	COMPLETADO	25/10/2018	42
5	Actualización de rutas	COMPLETADO	24/12/2018	41

Nota: Descripción del Sprint BackLog
Elaborado por: Sebastián Solís y Jonathan Almeida

2.3 Diagramas de casos de uso

Los diagramas de casos de uso sirven para tener una visión clara de los requerimientos y permiten conocer la interacción del funcionamiento con la aplicación móvil.

Tabla 8. Actores del sistema

ROL	TAREA
ADMINISTRADOR	Usuario con perfil de súper usuario y encargado de la inserción, modificación y eliminación de rutas de transporte. Adicional es el encargado de que el software mantenga un funcionamiento correcto.
USUARIO	El usuario es el responsable para la creación de una cuenta en el sistema para interactuar con los diferentes módulos de la aplicación móvil. Usa los módulos de búsqueda e inicio de sesión.

Nota: Descripción de los roles en la aplicación móvil
Elaborador por: Sebastián Solís y Jonathan Almeida

2.4 Búsqueda de rutas en DMQ

Proceso en el cual el usuario realiza la búsqueda de rutas por medio del ingreso del destino y la geolocalización del dispositivo móvil

Tabla 9. Caso de uso: búsqueda de rutas en DMQ

DESCRIPCIÓN CASO DE USO: BÚSQUEDA DE RUTAS EN DMQ	
ACTORES:	Usuario
OBJETIVO:	Establecer una conexión remota hacia el servidor para buscar las rutas de buses urbanos que sirvan para llegar hasta el destino ingresado.
DESCRIPCIÓN:	• Iniciar sesión con usuarios anteriormente creados. • Ingresar destino (lugares, nombres de calles) • Almacenar la información en coordenadas. • Buscar entre las rutas almacenadas en la base. • Mostrar en el Activity Maps las posibles rutas, trasbordos, paradas hasta la ruta de destino.

Nota: Descripción del proceso de búsqueda de rutas.
Elaborado por: Sebastián Solís y Jonathan Almeida

Caso de uso búsqueda rutas

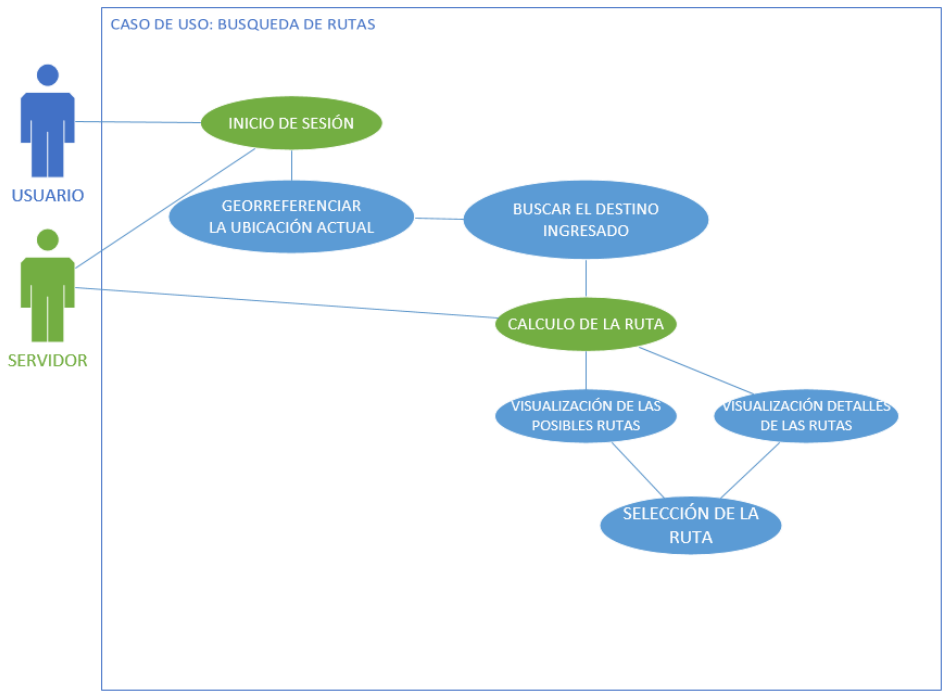


Figura 2. Descripción del proceso de búsqueda de rutas.
Elaborado por: Sebastián Solís y Jonathan Almeida

2.5 Ingreso de rutas

Los casos de uso que se describen a continuación permiten ingresar nuevas rutas en formato de coordenadas para su posterior almacenamiento en la base de datos y posterior procesamiento de información.

Tabla 10. Caso de uso: ingreso de nuevas rutas

DESCRIPCIÓN CASO DE USO: INGRESO DE NUEVAS RUTAS	
ACTORES:	Administrador
OBJETIVO:	Importar la información sobre la ruta en coordenadas para su almacenamiento en la base de datos con su identificador y códigos de ruta.

DESCRIPCIÓN:	• Importar información
	  ➢ Consultar información
	  o Conectar Servidor
	  ➢ Ingreso de información (coordenadas)
	  o Conectar servidor
	• Mostrar ruta ingresada

Nota: Descripción del proceso de ingreso de nuevas rutas.
Elaborado por: Sebastián Solís y Jonathan Almeida

Caso de uso Ingreso de nuevas rutas

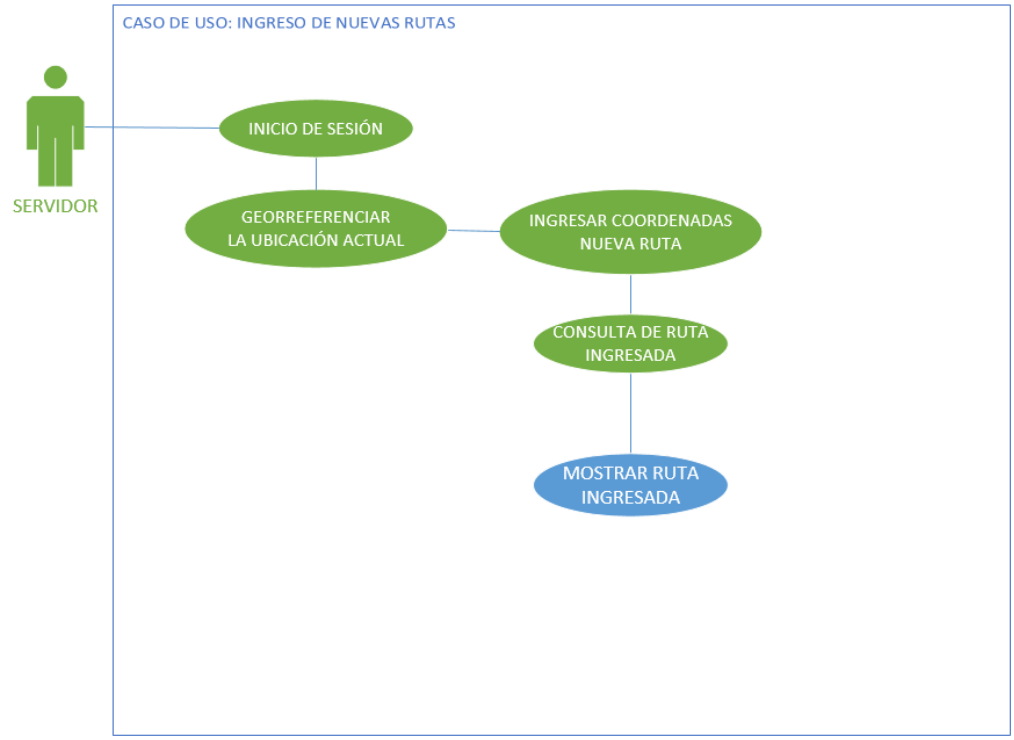


Figura 3. Descripción del proceso de ingreso de nuevas rutas.
Elaborado por: Sebastián Solís y Jonathan Almeida

2.6 Ingreso de nuevos usuarios

Los casos de uso que se describen a continuación permiten ingresar nuevos usuarios para poder hacer uso de la aplicación.

Tabla 11. Caso de uso: ingreso de nuevos usuarios

DESCRIPCIÓN CASO DE USO:	
ACTORES:	Administrador / Usuario
OBJETIVO:	Creación del usuario y contraseña para el acceso correcto a la aplicación.
DESCRIPCIÓN:	• Ingreso de correo electrónico • Ingreso de contraseña

Nota: Descripción del proceso de ingreso de nuevos usuarios.
Elaborado por: Sebastián Solís y Jonathan Almeida

Caso de uso Ingreso de nuevos usuarios

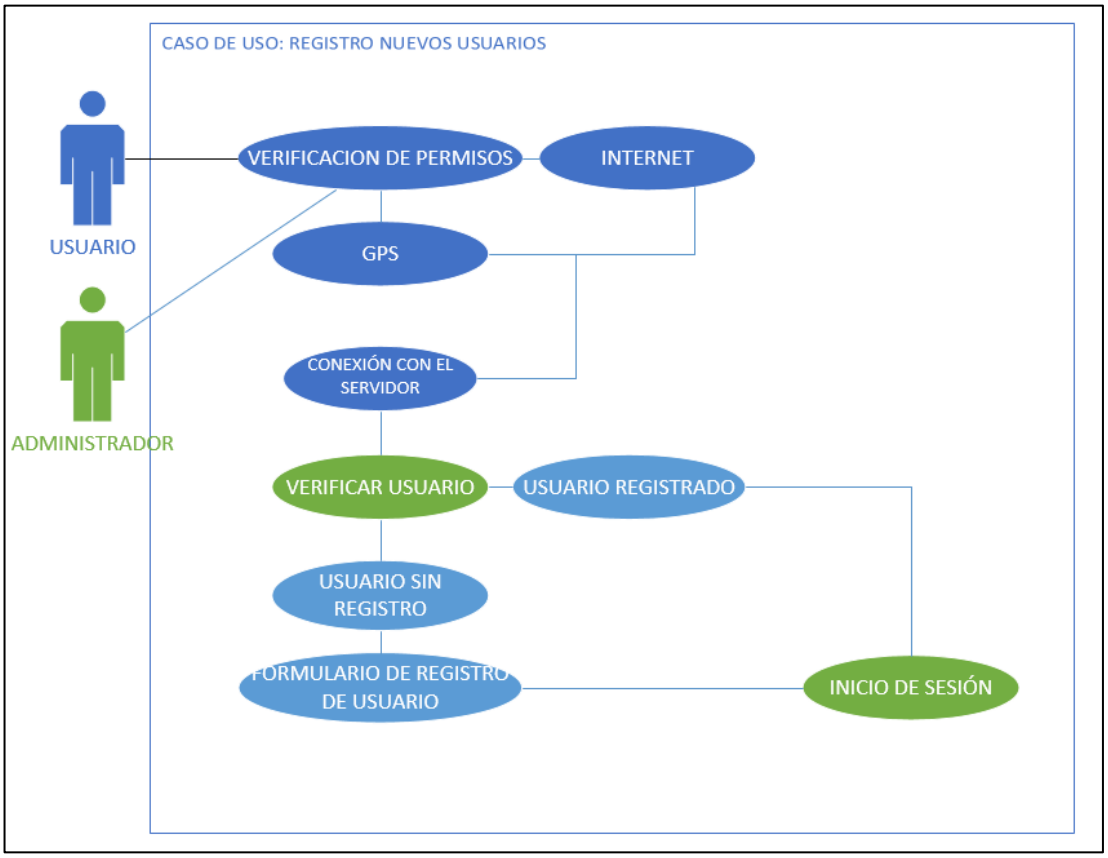


Figura 4. Descripción del proceso de ingreso de nuevos usuarios.
Elaborado por: Sebastián Solís y Jonathan Almeida

2.7 Diagrama de secuencia

Los diagramas que muestran la secuencia permiten tener una visión más clara de los sistemas divididos en módulos para lo cual se detallan a continuación:

El diagrama de secuencia de la figura No. 5 indica la autenticación de los usuarios registrados de la aplicación.

Autenticación de usuario

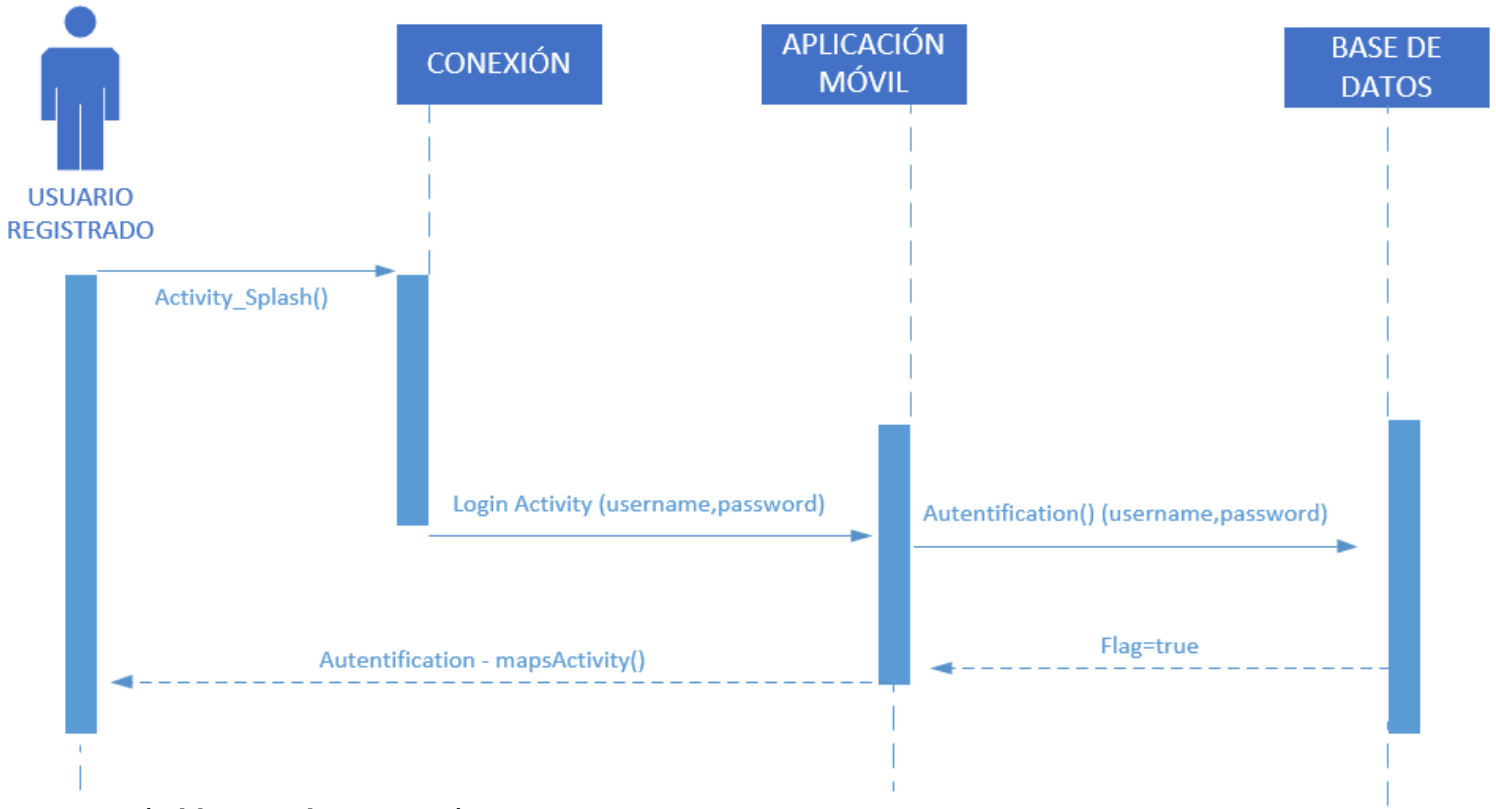


Figura 5. Descripción del proceso de autenticación.
Elaborado por: Sebastián Solís y Jonathan Almeida

El diagrama de secuencia de la figura No. 6 indica la autenticación de usuarios no registrados en la aplicación.

Autenticación de usuario no registrados en la aplicación

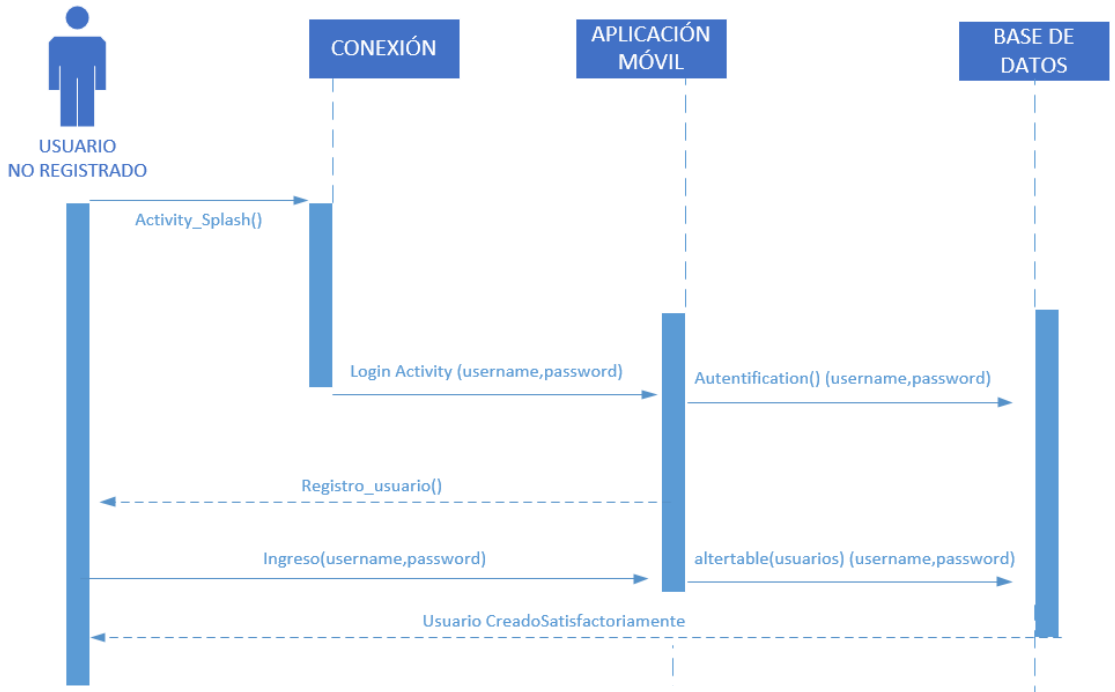


Figura 6. Descripción del proceso de autenticación de usuarios no registrados en la aplicación.
Elaborado por: Sebastián Solís y Jonathan Almeida

El diagrama de secuencia de la figura No. 7 indica el proceso de consulta y visualización de rutas del servidor de base de datos en la aplicación móvil.

Consulta y visualización de rutas

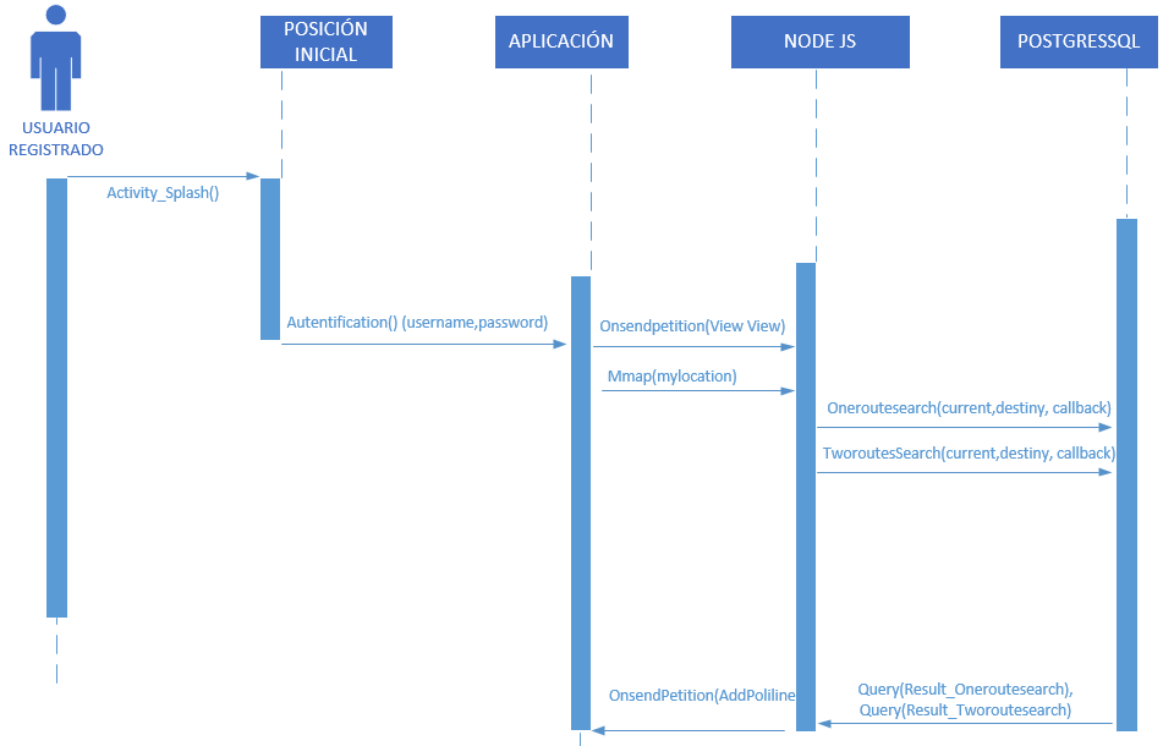


Figura 7. Descripción del proceso de consulta y visualización de rutas.
Elaborado por: Sebastián Solís y Jonathan Almeida

El diagrama de secuencia de la figura No. 8 indica la búsqueda de rutas en el api de Google Maps.

Búsqueda de rutas

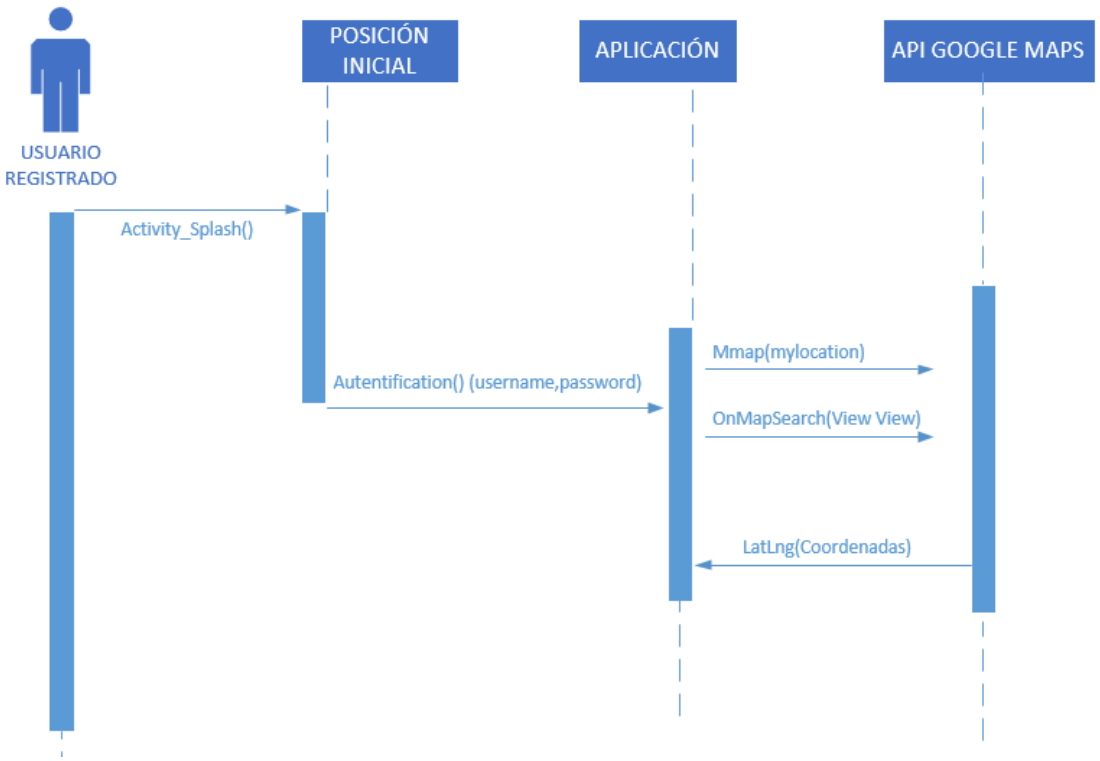


Figura 8. Descripción del proceso de búsqueda de rutas.
Elaborado por: Sebastián Solís y Jonathan Almeida

El diagrama de secuencia de la figura No. 9 indica el proceso de añadir nuevas rutas.

Añadir nuevas rutas

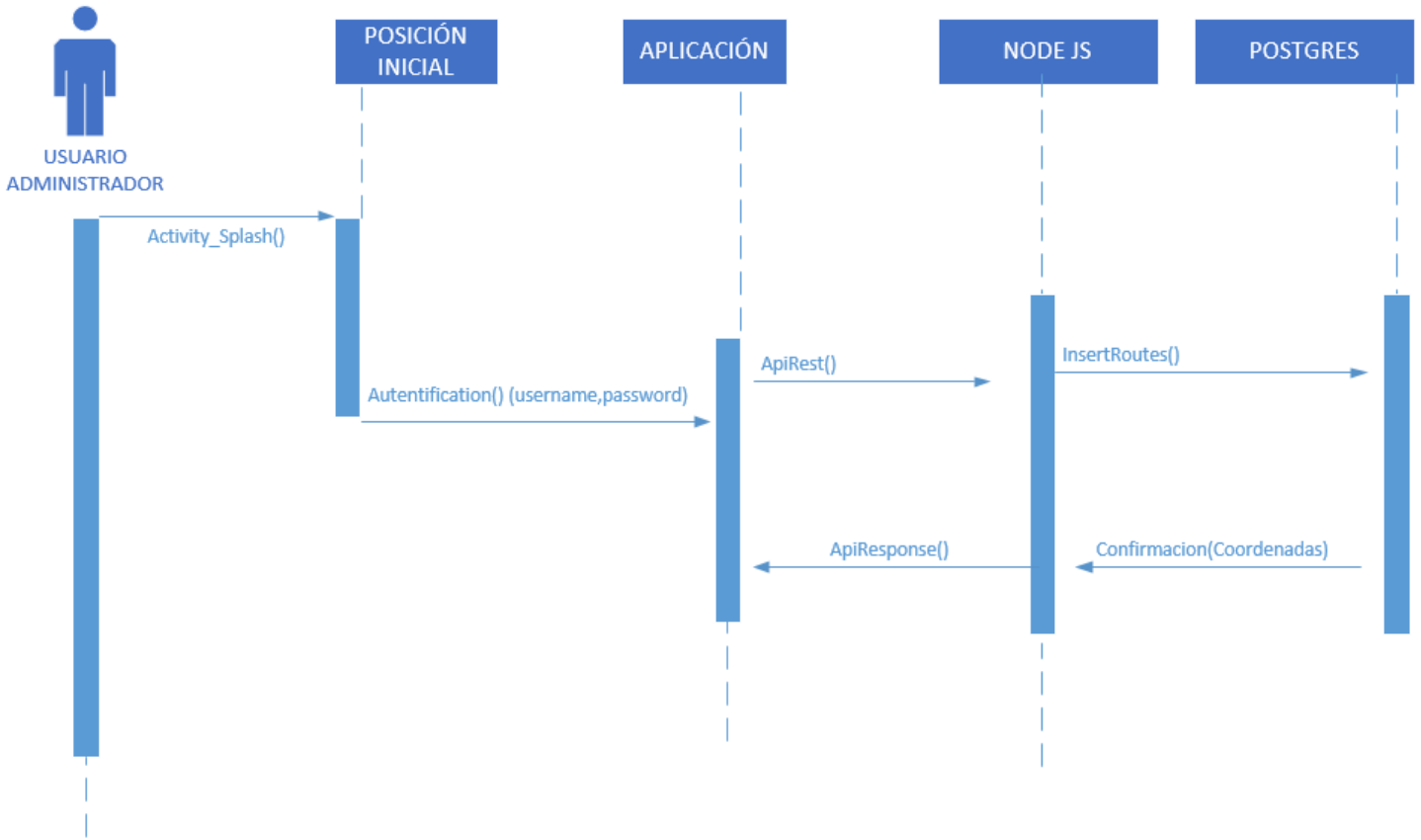


Figura 9. Descripción del proceso de añadir nuevas rutas
Elaborado por: Sebastián Solís y Jonathan Almeida

2.8 Modelo entidad relación

El diagrama físico de la base de datos que se muestra a continuación, el cual describe los aspectos relacionados con el modelado de datos en la base de datos.

Modelo entidad relación

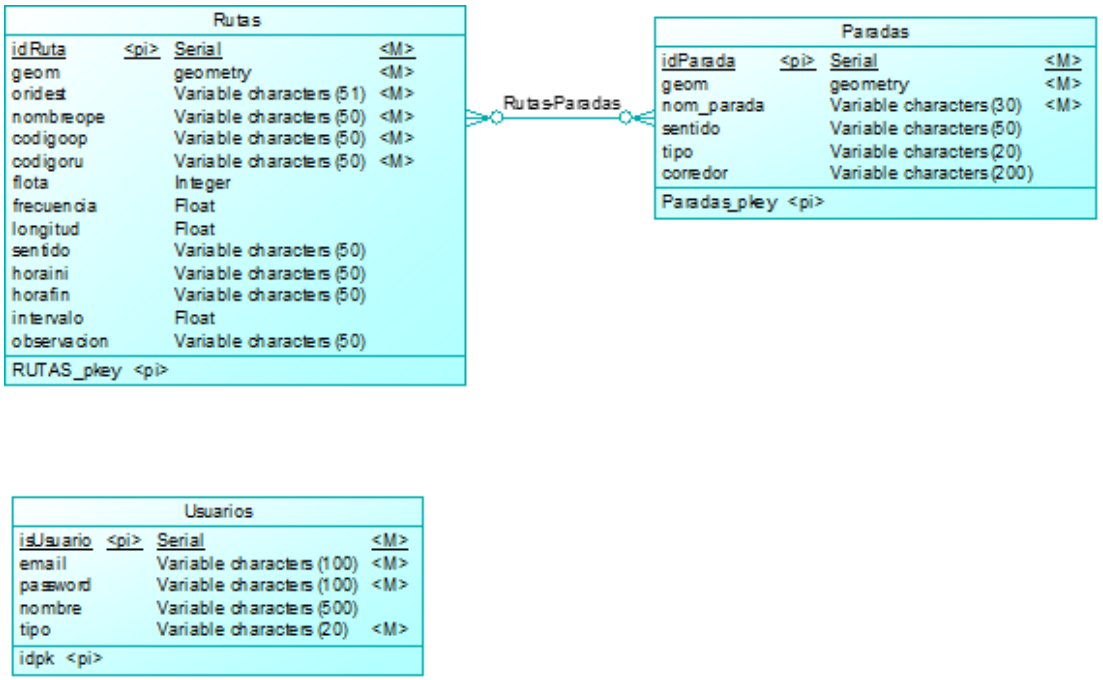


Figura 10. Modelo entidad relación de la base de datos.
Elaborado por: Sebastián Solís y Jonathan Almeida

Capítulo 3

Desarrollo y pruebas generales

3.1 Implementación

La implementación muestra los resultados del desarrollo de las tareas de cada sprint que se mencionaron en la tabla sprint backlog.

3.2 Implementación de primer sprint

Según las historias de usuario se establecieron para el primer sprint, se trabajaron las siguientes tareas:

3.3 Arquitectura del sistema

La arquitectura cuenta con tres capas: Presentación (aplicación Android), Negocio servidor web, api REST) y Acceso de datos (Postgis).

Esquema general

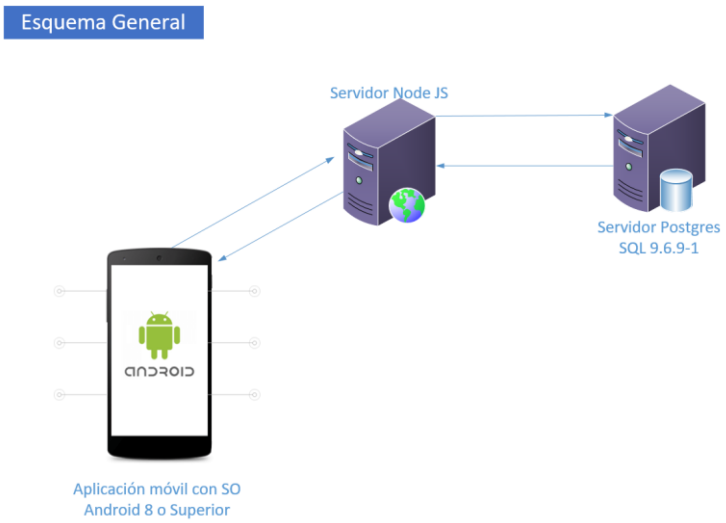


Figura 11. Modelo esquema general de la arquitectura del sistema.
Elaborado por: Sebastián Solís y Jonathan Almeida

3.3.1 Capa de presentación

Esta capa cuenta con la aplicación Android en donde está la interfaz con el usuario final permitiendo visualizar las rutas consultadas en el mapa, tomando data del GPS y búsqueda de lugares para la búsqueda de las mismas.

3.3.2 Capa de negocio

En la capa de negocio está el servidor web, la cual maneja la lógica y las funciones de la aplicación a través de un api REST.

Aquí se realizan los cálculos de rutas a través de scripts que usan funciones Postgis

Implementación del primer sprint

La función principal del proyecto es dar al usuario las rutas más convenientes hacia su destino.

3.3.3 Levantamiento del api y autenticación web

La funcionalidad del api web permite conexión a la base de datos realizando así autenticación, consultar o buscar las rutas, insertar nuevas rutas, eliminar rutas, o actualizarlas. Insertar, actualizar y eliminar necesita autenticación de usuarios administradores.

3.3.4 Creación servicio web

El servidor web utiliza el lenguaje de programación Node js, que permite comunicarse con la base de datos (Postgres Postgis) y ejecutar las consultas y modificaciones que permite este api. El archivo de configuración package.json como en la figura No. 12 tiene las versiones de Node js y sus módulos que se utilizaron.

Creación del servicio web

```
{
  "name": "servidor",
  "version": "1.0.0",
  "description": "Servidor web node js para manejar conexion con la base de datos",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Sebastian Solis - Jonathan Almeida",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.18.2",
    "express": "^4.16.3",
    "pg": "^7.4.3"
  }
}
```

Figura 12. Configuración de paquete json
Elaborado por: Sebastián Solís y Jonathan Almeida

En la figura No. 13 se importan los módulos o librerías que se utilizan para el uso de infraestructura web express, los objetos json (java object) a través de body-parser que traduce del formato x-www-form-urlencoded a json, y la conexión con postgrestsql (base de datos).

Importación de módulos

```
var express = require('express');
var app = express();
var bodyParser = require('body-parser');

const { Pool, Client } = require('pg');
//var client = new Client();

// parse application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({extended: false}));

// parse application/json
app.use(bodyParser.json());
```

Figura 13. Modelo de la importación de módulos
Elaborado por: Sebastián Solís y Jonathan Almeida

3.3.5 Desarrollo de autenticación web

Con la conexión ya realizada, se arma el api REST para la conexión externa (peticiones REST, aplicación móvil), comenzando con la autenticación contra la base de datos por medio del usuario (email) y la contraseña véase en la figura No. 14.

Autenticación en la base de datos

```
app.post('/', function (req, res) {
    var email = req.body.email;
    var pass = req.body.password;
    autenticacion(email, pass, "1,2", function(baut){
        if (baut) {
            res.send({"response":true});
        }else{
            res.send({"response":false});
        }
    });
});
```

Figura 14. Autenticación a nivel de base de datos
Elaborado por: Sebastián Solís y Jonathan Almeida

En la figura No. 15 se crea la función de autenticación para que sirva a todo el api, a través de una consulta a la tabla de usuarios, la respuesta se toma para determinar si el usuario existe y la contraseña es correcta, regresando un booleano para las decisiones en el api.

Función autenticación

```
function autenticacion(email, password, tipo, Callback){
    var query =
        'SELECT '+
            'nombre '+
        'FROM '+
            'esquema1.usuarios '+
        'WHERE '+
            "email = '"+email+"' and password = '"+password+"' and tipo in ("+tipo+") ";
    client.query(query, (err, res) => {
        console.log(err, res);
        if (res.rowCount > 0) {
            Callback(true);
        }else{
            Callback(false);
        }
    });
}
```

Figura 15. Función de autenticación que autoriza el inicio de sesión.
Elaborado por: Sebastián Solís y Jonathan Almeida

La respuesta a la petición POST de autenticación envía un booleano a través de un objeto json “response”: true (la contraseña y el usuario son correctos), “response”: false (el usuario o la contraseña son incorrectos).

3.3.6 Cálculo de rutas

El cálculo de rutas parte de la funcionalidad dentro del api web, de buscar y calcular las rutas necesarias para llegar al destino indicado. Para ello se desarrolla el módulo contando con: El servicio web, la funcionalidad de Búsqueda, el script de consulta.

3.3.7 Funcionalidad api web

En la figura No. 16 se desarrolla la funcionalidad del servicio web de búsqueda de rutas que se utilizara para la comunicación entre la aplicación móvil Android y el servidor.

Servicio web de búsqueda de rutas.

```
app.post('/SearchRoute', function (req, res) {
    var jsonReq = req.body;
    var current = jsonReq.current;
    var destiny = jsonReq.destiny;
    console.log(current);
    console.log(destiny);
    OneRouteSearch(current, destiny, function(closestP){
        console.log(closestP);
        if (closestP) {
            res.send({num:1, data: closestP});
        }else{
            TwoRoutesSearch(current, destiny, function(closestP1){
                if (closestP1) {
                    console.log("---");
                    res.send({num:2, data: closestP1});
                }else{
                    console.log("No se encontraron rutas");
                    res.send({num:0});
                }
            });
        }
    });
});
```

Figura 16. Servicio web para la búsqueda de rutas.
Elaborado por: Sebastián Solís y Jonathan Almeida

Se ande un servicio web /SearchRoute que recibe un objeto json de la petición post con los puntos referenciales (origen, destino) en un arreglo de coordenadas WGS 84 (longlat). Dirigiendo hacia las funciones que realizan el cálculo.

3.3.8 Desarrollo del script de consulta de rutas

Se desarrolla la funcionalidad del cálculo de las rutas a través de un proceso, que mediante dos scripts, basados en funciones PostGis, calcula la nueva geometría que es la ruta que retorna el servicio.

En el siguiente ejemplo de la figura No. 17 se detalla el script para la búsqueda de una ruta que conecte los puntos de origen y destino:

Script para la búsqueda de ruta

```
-- Obtenemos las Rutas que pasan por la parada origen
SELECT
	foo3.id
	,foo3.origen___d as nombre
	,sentido
	,foo2.Parada0 as Parada0 -- Parada
	,foo3.geom as geometria -- Ruta
	,ST_ClosestPoint(foo3.geom, foo2.Parada0) AS cPoint  -- Punto mas cercano a la parada dentro de la ruta
	,foo3."código_ru" as codRu
FROM
	(
		-- Obtenemos la parada mas cercana al punto recibido del Origen
		SELECT
			id
			,geom as Parada0
			,ST_Distance(ST_ClosestPoint(geom, pointCurrent), pointCurrent, true)
		FROM
			(
				-- Convertimos las cordenadas Lon Lat que recibimos de la aplicacion Android a Geometria
				SELECT
					ST_SetSRID(ST_MakePoint(-78.492265, -0.206165),4326) as pointCurrent
			)as foo,
			esquema1."PARADAS" -- Utilizamos las tablas con las paradas necesarias
		ORDER BY 3 ASC -- Ordenamos por la distancia mas corta
		LIMIT 1
	) AS foo2,
	esquema1."RUTAS" AS foo3, -- Utilizamos las Tablas con las Rutas requeridas
	esquema1."Paradas-Rutas" AS foo4
WHERE
	foo4."Paradas_id" = foo2.id
	and
	foo4."Rutas_id" = foo3.id
```

Figura 17. Script para la búsqueda de ruta
Elaborado por: Sebastián Solís y Jonathan Almeida

Se obtiene la geometría de la nueva ruta basada en la combinación de las rutas que pasan por el punto origen y el punto destino, se describe en la figura No.18.

Obtener nueva ruta

```
--calculo de la Ruta Final
,ST_GeometryN(
        -- Se divide la Ruta en dos partes y se conserva la primera
        ST_Split(
                -- Ruta que pasa por la parada cercana al Origen
                ST_GeometryN(
                -- division de la Ruta Origen en dos partes conservando la segunda
                        ST_Split(
                                Origin.geometria
                                , ST_SetSRID(
                                        ST_GeomFromText(
                                                'LINESTRING(' || ST_X(Origin.cPoint) || ' ' || ST_Y(Origin.cPoint) + 0.00001
                                                || ', ' || ST_X(Origin.cPoint) || ' ' || ST_Y(Origin.cPoint) - 0.00001 || ')'
                                        )
                                        , 4326
                                )
                        )
                        , 2
                )
                -- Ruta que pasa por la parada cercana al Destino
                , ST_SetSRID(
                        ST_GeomFromText(
                                'LINESTRING(' || ST_X(Destiny.cPoint) || ' ' || ST_Y(Destiny.cPoint) + 0.00001
                                || ', ' || ST_X(Destiny.cPoint) || ' ' || ST_Y(Destiny.cPoint) - 0.00001 || ')'
                        )
                        , 4326
                )
        )
        , 1
) as RouteFinale
```

Figura 18. Obtener nueva ruta según la combinación de rutas
Elaborado por: Sebastián Solís y Jonathan Almeida

Aumentando una función para calcular la longitud de la nueva ruta se puede ordenar los resultados para que se devuelva la ruta más corta las condiciones para unir las dos rutas como se muestra en la figura No.19.

Condiciones para obtención de la ruta más corta

```
WHERE
-- Tomamos las rutas que pasan por la parada origen y por la parada destino
        Origin.id = Destiny.id
        and Origin.codRu = Destiny.codRu
-- Tomamos solamente las rutas que van en el sentido Origen - Destino
-- dividimos la ruta en dos partes por la parada origen, y si en la segunda parte esta la parada destino cojemos esas Rutas
        and ST_DWithin(
                Destiny.cPoint
                , ST_GeometryN(
                        ST_Split(
                                ST_LineMerge(
                                        Origin.geometria
                                )
                                , ST_SetSRID(
                                        ST_GeomFromText(
                                                'LINESTRING(' || ST_X(Origin.cPoint) || ' ' || ST_Y(Origin.cPoint) + 0.00001
                                                || ', ' || ST_X(Origin.cPoint) || ' ' || ST_Y(Origin.cPoint) - 0.00001 || ')'
                                        )
                                        , 4326
                                )
                        )
                        , 2
                )
                , 5
                , true
        ) = 't'
ORDER BY 13 ASC
```

Figura 19. Consulta para determinar la ruta más corta
Elaborado por: Sebastián Solís y Jonathan Almeida

La respuesta a la petición se organiza para enviar la información y las coordenadas de las rutas en un json en la disposición dada en la figura No. 20.

Petición de coordenadas en formato json

```
client.query(query, (err, res) => {
    console.log(res.rowCount);
    if (res.rowCount > 0) {
        var ParadaOrigen = JSON.parse(res.rows[0].originstop).coordinates[0];
        var Ruta = JSON.parse(res.rows[0].route).coordinates;
        var ParadaDestino = JSON.parse(res.rows[0].destinystop).coordinates[0];
        //armando el json de retorno
        var jsonPOri = {latitude:parseFloat(ParadaOrigen[1]).toFixed(7), longitude: parseFloat(ParadaOrigen[0]).toFixed(7)};

        var jsonRoute = [{latitude:parseFloat(ParadaOrigen[1]).toFixed(7), longitude: parseFloat(ParadaOrigen[0]).toFixed(7)}];
        for (var i = 0; i < Ruta.length; i++) {
            jsonRoute.push({latitude:parseFloat(Ruta[i][1]).toFixed(7), longitude: parseFloat(Ruta[i][0]).toFixed(7)});
        }
        jsonRoute.push({latitude:parseFloat(ParadaDestino[1]).toFixed(7), longitude: parseFloat(ParadaDestino[0]).toFixed(7)});

        var jsonPDes = {latitude:parseFloat(ParadaDestino[1]).toFixed(7), longitude: parseFloat(ParadaDestino[0]).toFixed(7)};
        //enviando el json de retorno
        Callback({pOri: jsonPOri, pDes: jsonPDes, route: jsonRoute});
    }else{
        Callback(null);
    }
});
```

Figura 20. Petición de coordenadas en formato json
Elaborado por: Sebastián Solís y Jonathan Almeida

3.4 Implementación del segundo sprint

Este sprint contiene el desarrollo de la aplicación móvil para representar gráficamente y mediante una interfaz la búsqueda de rutas.

3.4.1 Aplicación móvil

La aplicación móvil está desarrollada para dispositivos Android, mediante el IDE Android Studio. Se utiliza el servicio web para la autenticación y la consulta de rutas en scripts que se ejecutan en la base de datos (Postgres)

Los servicios web que sirven como intermedio para la comunicación con la aplicación móvil son:

La aplicación tiene la siguiente estructura

Autenticación

Para acceder al servicio web se requiere una autenticación contra la base de datos, la misma se realiza a través del servicio enviando una petición post a /

Autenticación de la aplicación móvil

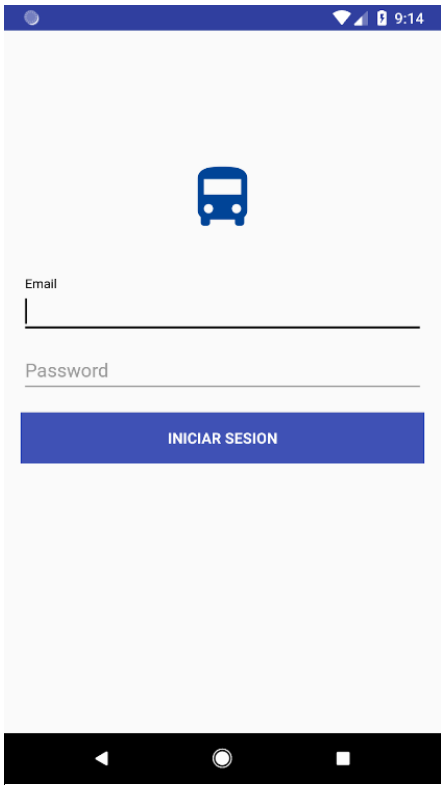


Figura 21. Inicio de sesión en la aplicación
Elaborado por: Sebastián Solís y Jonathan Almeida

Búsqueda de direcciones y lugares

Se crean marcadores para establecer los puntos de origen y destino en el mapa, dados por el GPS (punto actual del usuario) y una búsqueda en el api de google para encontrar los lugares delimitados para Quito-Ecuador o la creación manual del marcador del destino.

Búsqueda de direcciones y lugares por marcadores

Figura 22. Búsqueda de direcciones a través de marcadores en el mapa.
Elaborado por: Sebastián Solís y Jonathan Almeida

En la figura No. 22 se muestra el mapa con las funcionalidades de la aplicación, marcando el destino por medio de la barra de búsquedas o haciendo un clic en el mapa

Búsqueda de direcciones y lugares en la barra de búsqueda

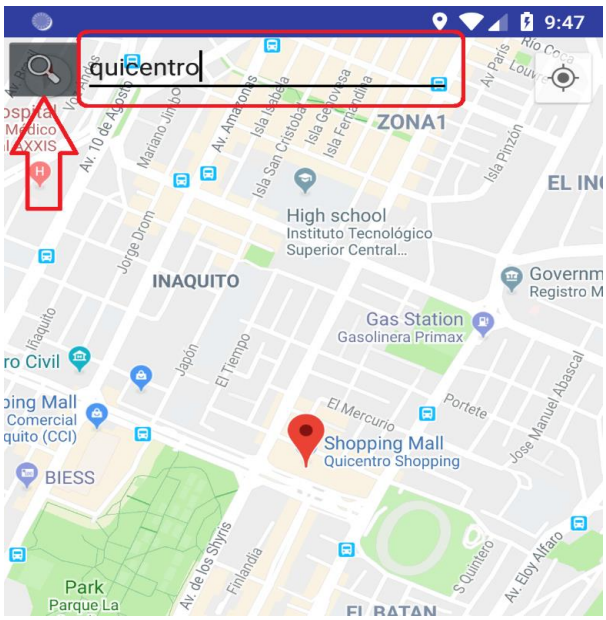


Figura 23. Búsqueda de direcciones y lugares a través de la barra de búsqueda.
Elaborado por: Sebastián Solís y Jonathan Almeida

Al usar la barra de búsqueda se obtendrá el marcador para realizar la consulta de rutas a ese destino véase en la figura No. 23

Consulta de rutas al servidor web

Se envía los datos requeridos al servidor web y se obtiene (en caso de encontrar) las rutas en un arreglo, que la aplicación convierte en polígonos que se visualizan en el mapa, junto con la información de la ruta.

Visualización de rutas

Figura 24. Visualización de rutas
Elaborado por: Sebastián Solís y Jonathan Almeida

Administración

El módulo permite al administrador gestionar las operaciones que se realizan sobre la base de datos

3.5 Implementación del tercer sprint

El api web permite gestionar la base de datos, teniendo el administrador que autenticarse

3.5.1 Inserción de rutas

El servicio web para la inserción de rutas da acceso a crear un nuevo registro en la tabla rutas, tomando en cuenta esta nueva geometría para cálculos posteriores.

Desarrollo del servicio web

Se desarrolla el servicio web que permite la inserción de una nueva ruta, el administrador tiene que realizar una petición Post al api /InsertRoute enviando el siguiente objeto json en la figura No. 25.

Json para insertar nueva ruta

Figura 25. Json para insertar una nueva ruta
Elaborado por: Sebastián Solís y Jonathan Almeida

Funcionalidad de inserción de rutas

Para la inserción de nuevas rutas después se arma el string con los datos que se obtiene del json obtenido de la petición post, separando la información, y de la geometría de la ruta (array de puntos representados en coordenadas lonlat) véase el código en la figura No 26.

Funcionalidad de inserción rutas

```
app.post('/InsertRoute', function (req, res) {
    var email = req.body.email;
    var pass = req.body.password;
    autenticacion(email, pass, 1, function(baut){
        if (baut) {
            var arrayGeom = req.body.geom;
            var data = req.body.info;
            var geom = 'MULTILINESTRING((';
            for (var i = 0; i < arrayGeom.length - 1; i++) {
                geom = geom + arrayGeom[i][0] +" "+ arrayGeom[i][1]+", ";
            }
            geom = geom + arrayGeom[arrayGeom.length - 1][0] +" "+ arrayGeom[arrayGeom.length - 1][1]+"))";
            console.log(geom);
            InsertRoute(geom, data, function(jSonResp){
                res.send(jSonResp);
            });
        }else{
            res.send({error: true, msj: "email o password incorrectos"});
        }
    });
});
```

Figura 26. Json para insertar una nueva ruta
Elaborado por: Sebastián Solís y Jonathan Almeida

Se llama a la función insertar (figura No. 26) ruta en donde se arma el query de inserción con la transformación de coordenadas al sistema 4326 del sistema de coordenadas lonlat de la geometría.

Transformación de coordenadas

```
INSERT INTO esquema1.rutas(geom, oridest, nombreope, codigoop,
codigoru, flota, frecuencia, longitud, sentido, horaini, horafin, intervalo,
observacion)
```

Figura 27. Sentencia para la transformación de coordenadas
Elaborado por: Sebastián Solís y Jonathan Almeida

El resultado del script se devuelve en un booleano y la información de la operación dentro de un objeto Json de la figura No. 28.

Resultado de la operación de transformación

```
client.query(query, (err, res) => {
    //console.log(err, res);
    if (err) {
        Callback({error:true, info: err});
    }else if(res.rowCount=1){
        Callback({error:false, info: res});
    }else{
        Callback({error:true, info: res});
    }
});
```

Figura 28. Resultado de la operación de transformación
Elaborado por: Sebastián Solís y Jonathan Almeida

Para ejecutar el script se necesita los datos del administrador, junto con la información necesaria de la ruta y un arreglo con las coordenadas lonlat que creará la geometría en Postgres.

3.5.2 Eliminación de rutas

Permite la eliminación de los registros indicados incluyendo las relaciones existentes que sirven para el cálculo de rutas.

Desarrollo del servicio web

Para la eliminación de rutas se desarrolla el servicio web que permite peticiones post a /DeleteRoute enviando el siguiente json de la figura No. 29.

Desarrollo del servicio web para eliminar ruta

```
{
    "email":"admin"
    ,"password":"adminRutasUIO"
    ,"id": 2
}
```

Figura 29. Desarrollo del servicio web para eliminar rutas.
Elaborado por: Sebastián Solís y Jonathan Almeida

Funcionalidad de eliminación de rutas

El script de eliminación utiliza el id de la ruta a eliminar, además de los datos del administrador para la autenticación como se observa en la figura No. 30:

Codificación para eliminar una ruta

```
//Delete a route with the id
app.post('/DeleteRoutePrueba', function (req, res) {
    var email = req.body.email;
    var pass = req.body.password;
    var id = req.body.id;
    if (typeof id !== "undefined") {
        autenticacion(email, pass, 1, function(baut){
            if (baut) {
                client.query("DELETE FROM esquema1.rutas WHERE id = "+id, (err, res) => {
                    if (err) {
                        res.send({error:true, info: err, msj: String(err)});
                    }else if(res.rowCount=1){
                        res.send({error:false, info: res});
                    }else{
                        res.send({error:true, info: res, msj: "No se modifico ninguna fila"});
                    }
                });
            }else{
                res.send({error: true, msj: "email o password incorrectos"});
            }
        });
    }else{
        res.send({error: true, msj: "Especificar el id de la Ruta"});
    }
});
```

Figura 30. Desarrollo del servicio web para eliminar rutas.
Elaborado por: Sebastián Solís y Jonathan Almeida

El servicio responde con un json que informa si se realizó la inserción o si hubo errores en el proceso.

3.5.3 Actualización de rutas

Realiza la modificación de los registros indicando los parámetros requeridos.

Desarrollo del servicio web

En la actualización de rutas se necesita la petición Post a /UpdateRoute enviando el siguiente json de la figura No. 31.

Desarrollo del servicio web para actualización


```
{
    "email":"admin"
    ,"password":"adminRutasUIO"
    ,"id": 2
    ,"info":{
        "oridest": "pepito"
        ,"nombreOpe": "juanito"
        ,"codigoOp": "juan"
        ,"codigoRu":"23"
        ,"flota": 13
        ,"frecuencia": 23
        ,"longitud": 23.34
        ,"sentido": "Ida"
        ,"horaIni": "5:15:00"
        ,"horaFin": "21:00:00"
        ,"intervalo": 10
        ,"observacion": "Informacion adicional"
    }
}
```


Figura 31. Desarrollo del servicio web para actualizar rutas.
Elaborado por: Sebastián Solís y Jonathan Almeida

Para ejecutar el script es mandatorio incluir la información del administrador, como el id de la ruta que se requiere modificar, no es necesario enviar todos los parámetros que tiene la ruta solamente los que se quiera actualizar, incluyendo el array con las coordenadas (figura No. 32) si se quiere actualizar la geometría.

Actualización de la geometría


```
,"geom":[
    [-78.558631033125,-0.324834247406326],[-78.5585550738258,-0.325124437506924],[
        .5584820724604,-0.325494432038787],[-78.558420072211,-0.325766426774034],[
        .5583900723509,-0.325885425089816],[-78.5582203330128,-0.326559594621452],
        .5582130707059,-0.3265884130208],[-78.5583040693402,-0.32650141517892],[-7
        .5586060657761,-0.326367418694023],[-78.5586410651638,-0.326347419016147],
        .5603330030604,-0.332337334403541],[-78.5601590058591,-0.332339333302823],
        .5599710088784,-0.332342332103536],[-78.5597290127713,-0.332345330578879],
        .5594980164518,-0.332354329029871],[-78.5593730185745,-0.332337328547529],[
        .5590280240946,-0.332347326304247],[-78.5587020293101,-0.332357324186584],
        .5585690314862,-0.332353323456453],[-78.5581030389603,-0.332365320491422],
        .5576520462661,-0.332365317830939],[-78.5571590543079,-0.332356315092255],
        .5567790606521,-0.332324313404742],[-78.5567180616696,-0.332319313133027],
        .5561050717987,-0.332286310144269],[-78.5557190781792,-0.332265308285826],
        .5556880791749,-0.332177309537434],[-78.5556310803621,-0.332130309976419],
        .5555030826608,-0.332090309900563],[-78.5553690850671,-0.332048309824967],
        .5550800903115,-0.331947309837074],[-78.5547310959023,-0.331959307691901],
        .5543941004323,-0.332130303045303]
    ]
```


Figura 32. Actualización de la geometría.
Elaborado por: Sebastián Solís y Jonathan Almeida

Funcionalidad de actualización de rutas

El proceso para actualizar rutas (figura No. 33) genera el string (query) con los datos del json, de forma que el nombre del objeto json sea el nombre del parámetro de la tabla en la base de datos.

Generación del script en formato json para actualización

```
if (typeof req.body.geom !== "undefined") {
    var arrayGeom = req.body.geom;
    var geom = 'MULTILINESTRING((';
    for (var i = 0; i < arrayGeom.length - 1; i++) {
        geom = geom + arrayGeom[i][0] +" "+ arrayGeom[i][1]+", ";
    }
    geom = geom + arrayGeom[arrayGeom.length - 1][0] +" "+ arrayGeom[arrayGeom.length - 1][1]+"))";
    //query
    queryParams = queryParams + "geom = ST_SetSRID(ST_GeomFromText('" + geom + "'), 4326), ";
    for (var i = 0; i < parameters.length - 1; i++) {
        queryParams = queryParams + parameters[i] + " = '" + data[parameters[i]] + "', ";
    }
    queryParams = queryParams + parameters[parameters.length - 1] + " = '" + data[parameters[parameters.length - 1]] +

}else{
    for (var i = 0; i < parameters.length - 1; i++) {
        queryParams = queryParams + parameters[i] + " = '" + data[parameters[i]] + "', ";
    }
    queryParams = queryParams + parameters[parameters.length - 1] + " = '" + data[parameters[parameters.length - 1]] +
}
```

Figura 33. Generación del script en formato json para la actualización
Elaborado por: Sebastián Solís y Jonathan Almeida

3.6 Pruebas

3.6.1 Plan de pruebas

Con el fin de brindar un servicio estable y funcional se estable las pruebas del sistema, estas pruebas para que se puedan verificar, analizar y organizar la retroalimentación se sigue un plan de pruebas.

3.6.2 Pruebas funcionales

Las pruebas funcionales son pruebas de caja negra que mediante la revisión de los procesos del sistema se evalúan y verifica las funcionalidades de los requerimientos establecidos inicialmente en las tareas.

En la ejecución de las pruebas funcionales se evalúan con respecto a los casos de uso establecidos, y se desarrollaran con la tabla siguiente como estándar para el plan de pruebas:

Tabla 12. Descripción del plan de pruebas

MÓDULO	PRE REQUISITO	ACTOR	PROCESO	ACCIONES	RESULTADO ESPERADO	RESULTADO ALCANZADO
Administración del sistema	Primera petición para generar la conexión en el servidor.	Administrador/ Usuario Registrado	Levantamiento del api y autenticación	Creación del servicio web	Conexión al api web mostrando el mensaje de estado de conexión	OK
				Autenticación web	Verificación de usuario registrado en la base de datos	OK
	Autenticación mediante usuario y contraseña	Administrador/ Usuario Registrado	Búsqueda de ruta	Calcular ruta	Obtener la ruta calculada	OK
			Búsqueda de direcciones y lugares	Desarrollo de la búsqueda de lugares	Marcadores con las coordenadas de los lugares encontrados	OK
			Sugerencia de rutas del servidor web	Buscar rutas	Obtener la o las rutas calculadas	OK
		Administrador	Inserción de geometría	Desarrollo de la funcionalidad de inserción de rutas	Nuevo Registro en la tabla rutas y mensaje de respuesta json	OK
		Administrador	Eliminar geometría	Desarrollo de la funcionalidad de eliminación de rutas	Registro eliminado de la tabla rutas y mensaje de respuesta json	OK
		Administrador	Actualizar geometría	Desarrollo de la funcionalidad de actualización de rutas	Registro modificado de la tabla rutas y mensaje de respuesta json	OK

Nota: Descripción del plan de pruebas
Elaborado por: Sebastián Solís y Jonathan Almeida

3.6.3 Pruebas de rendimiento

Estas pruebas son ejecutadas con Restfull Stress 1.6.0 que permite medir mediante pruebas de estrés los servicios web del servidor, esta herramienta permite escoger el número de usuarios o iteraciones, el tiempo antes de realizar él envió (delay), el tiempo máximo de espera para la respuesta (timeout) de la prueba para simular un ambiente real de carga, en el caso de Restfull Stress también se establecen las peticiones de calentamiento para que los datos en el inicio de la prueba sean consistentes.

Las pruebas de rendimiento se establecieron para el requerimiento de búsqueda de rutas, que es el proceso crítico de la aplicación, siendo también al que todos los usuarios tienen acceso.

Esta prueba se realiza para el cálculo de una ruta en una red local, y los parámetros para la realizar la prueba son: iteraciones 500 (500 peticiones de usuario), delay 100ms, timeout 30s.

Pruebas de estrés de una ruta

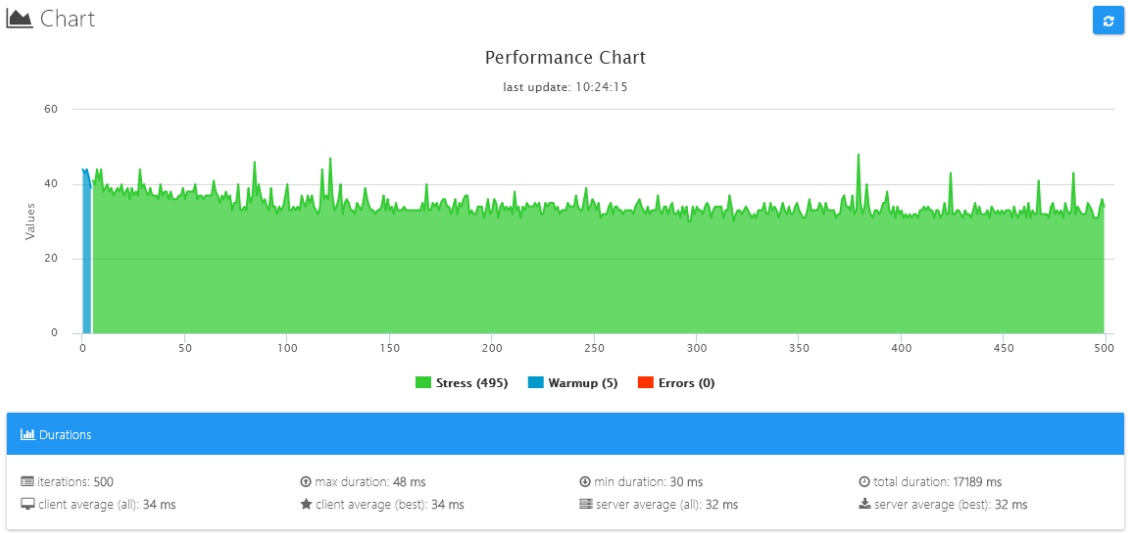
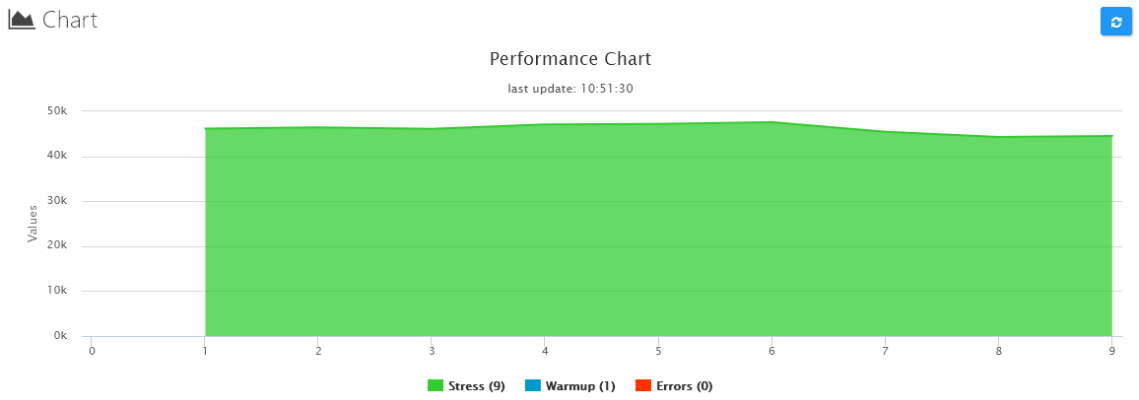


Figura 34. Pruebas de estrés en ejecución del script de búsqueda de una ruta.
Elaborado por: Sebastián Solís y Jonathan Almeida

En la figura No. 34 se visualiza que se ejecutó todas las peticiones sin errores con un tiempo promedio 34ms en el cliente y 32ms en el servidor.

Esta prueba se realiza para el cálculo de dos rutas en una red local, y los parámetros para la realizar la prueba son: iteraciones 10 (10 peticiones de usuario), delay 100ms, timeout 120.

Pruebas de estrés de dos rutas



XHR requests — HELP

detected requests/responses: 10 / 10 — requests for iteration: 1 — think time (ms): 5000

	client (all)	client (best)	server (all)	server (best)
average duration	46113 ms	45948 ms	46111 ms	45946 ms
throughput	0.02 req/sec	0.02 req/sec	0.02 req/sec	0.02 req/sec
inter-arrival time	51113 ms	50948 ms	51111 ms	50946 ms
arrival rate	0.0000195645	0.0000196279	0.0000195653	0.0000196286
concurrent users	1	1	1	1
parallel conc. users (+120% / +500%, observ.)	1 5	1 5	1 5	1 5

Figura 35. Pruebas de estrés en ejecución de script de búsqueda de dos rutas.
Elaborado por: Sebastián Solís y Jonathan Almeida

El rendimiento describe que las peticiones se ejecutan en un promedio de 51s sin errores como se observa en la figura No. 35.

3.6.4 Ambiente de pruebas

Con el fin de aprobar las pruebas, tanto como funcionales como no funcionales se toma en cuenta que estas se realizan en los siguientes ambientes:

El servidor web se ejecuta en la siguiente computadora:

Tabla 13. Especificaciones servidor web

Procesador	Memoria	Almacenamiento	Sistema Operativo	Ancho de banda
Intel core i7 4770k 3.5GHz	16 GB DDR3 1440 MHz	1 TB HDD 250 GB SSD	Windows 10 Pro	20 Mbps simétricos

Nota: Especificaciones del servidor web
Elaborado por: Sebastián Solís y Jonathan Almeida

El dispositivo móvil cuenta con las siguientes especificaciones:

Tabla 14. Especificaciones del dispositivo móvil

Dispositivo	Procesador	Memoria	Versión Android
Oneplus 3t	Snapdragon 821 2.35GHz	6 GB LPDDR4	8.0 Oreo

Nota: Especificaciones del dispositivo móvil
Elaborado por: Sebastián Solís y Jonathan Almeida

Las pruebas se realizaron por medio de una red local para el ambiente de desarrollo.

CONCLUSIONES

- La aplicación móvil facilita la visualización de diferentes rutas que atraviesan El Distrito Metropolitano de Quito de una forma visual, intuitiva y clara en un mapa mostrando información de las líneas de buses de transporte público.
- El uso del Api Web simplifica la gestión de la base de datos dado que al estar predeterminada la funcionalidad se automatiza el uso de scripts en el lenguaje estructurado de consultas (SQL).
- La metodología Scrum ayuda al desarrollador a poner en marcha las distintas actividades del proyecto teniendo la flexibilidad para realizar cambios según los requerimientos del usuario.
- El uso de las funciones PostGis permite construir con facilidad las geometrías (rutas) proporcionando las herramientas adecuadas para solventar los desafíos que se generan al trabajar con bases de datos espaciales.

RECOMENDACIONES

- Para trabajar de mejor manera con bases de datos espaciales se recomienda el uso de sistemas de coordenadas estándares como lo es el WGS-84, ya que al tener un sistema de coordenadas distinto requiere que sus archivos de especificaciones se transformen para poder ser interpretados y utilizados para el desarrollo de aplicaciones.
- Precalcular las relaciones entre las rutas y paradas permite optimizar los tiempos de ejecución manteniendo fuera el cálculo de pertenencia entre rutas y paradas.
- Para la estabilidad del desarrollo se recomienda mantener las versiones del Framework, base de datos, servidores durante todo el proceso de programación del proyecto.
- El uso de QGis ayuda a interpretar visualmente las rutas durante el proceso de diseño, ejecución y pruebas de los scripts.

LISTA DE REFERENCIAS

Booch, G. R. (1999). *El lenguaje unificado de modelado .* Madrid: Addison wesley.

Cantelon, M. H. (2014). *Node. js in Action.* Greenwich: Manning.

Chang, K. T. (2006). *Introduction to geographic information system.* Boston: McGraw-Hill Higher Education.

Connolly, T. M. (2005). *Database systems: a practical approach to design, implementation, and management.* Pearson Education.

Doglio, F. (2015). *Pro REST API Development with Node. js.* Apress.

Flanagan, D. (2006). *JavaScript: the definitive guide.* O'Reilly Media, Inc.

Gironés, J. T. (2012). *El gran libro de Android.* Marcombo.

Hohensee, B. (2014). *ntroducción A Android Studio. Incluye Proyectos Reales Y El Código Fuente.* Babelcube Inc.

Hugentobler, M. (2008). *Quantum gis. In Encyclopedia of GIS.* Springer, Boston, MA.

IBM. (s.f.). *IBM knowledgecenter*. Obtenido de https://www.ibm.com/support/knowledgecenter/es/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ac55710_.htm

Larman, C. (2003). *UML y Patrones.* Madrid: Pearson Educación.

Open, B. (02 de 2019). *BBVA Open 4u*. Obtenido de https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

Ramsey, P. (2005). *Postgis manual.* Refractions Research Inc.

Schwaber, K. &. (2002). *Agile software development with Scrum (Vol. 1).* Prentice Hall.

Techlandia. (02 de 2019). *Techlandia*. Obtenido de https://techlandia.com/son-bases-datos-espaciales-info_254596/